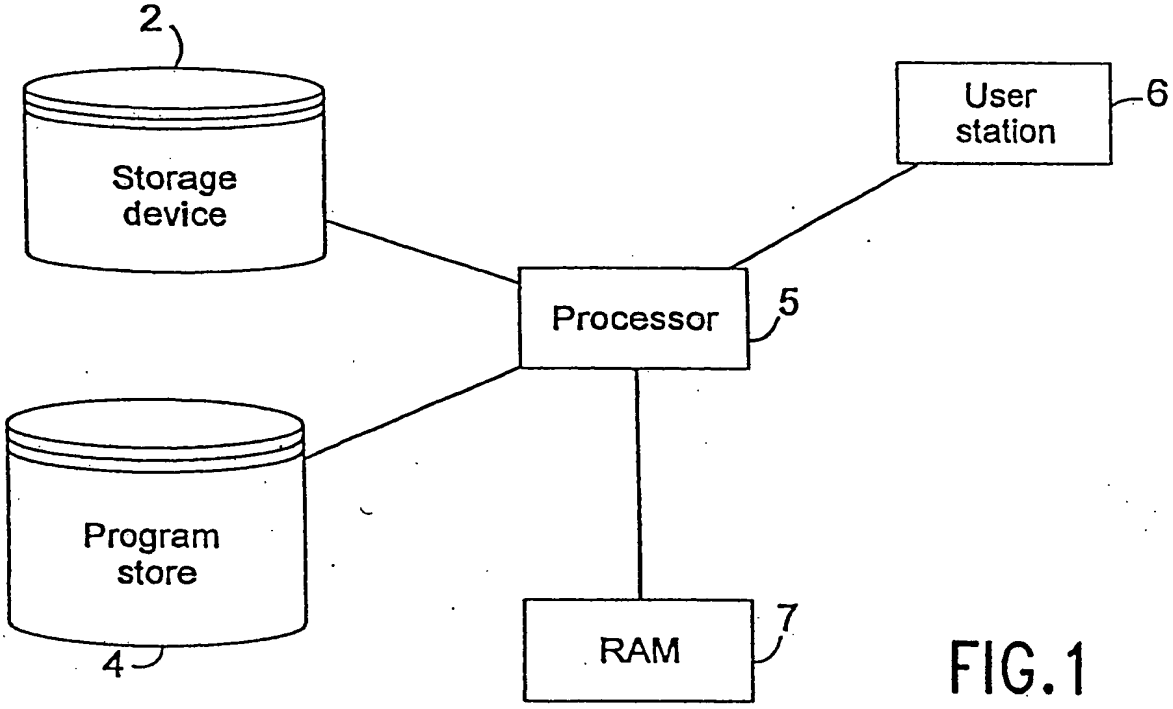
2
Storage device
User station
6
Processor
5
Program store
4
RAM
7

FIG.1

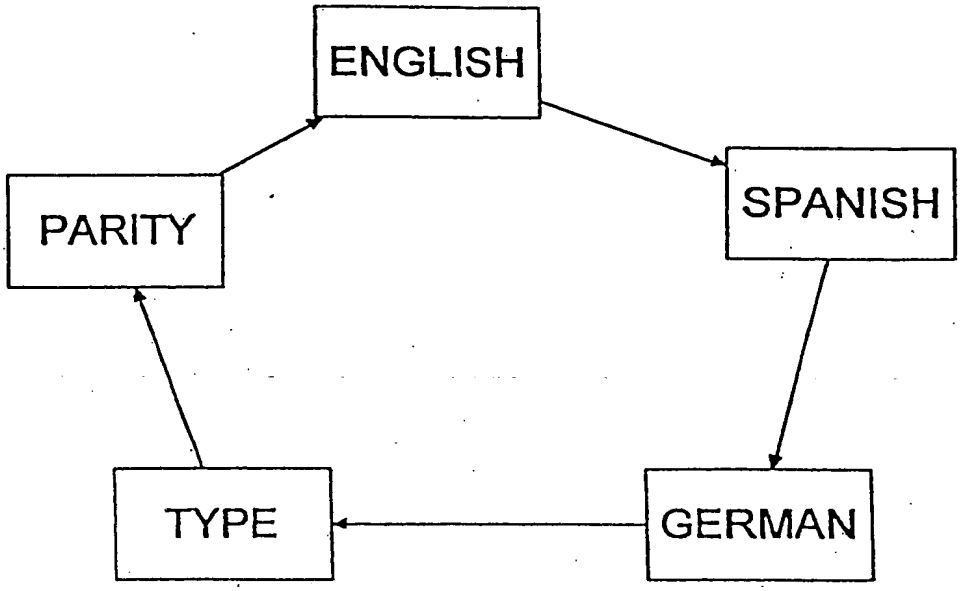
ENGLISH
SPANISH
PARITY
TYPE
GERMAN

FIG.2

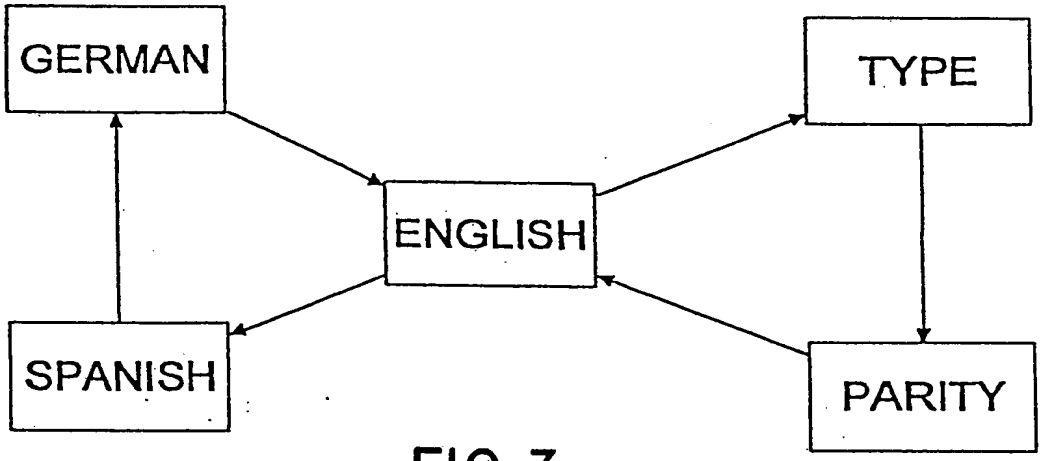

FIG.3

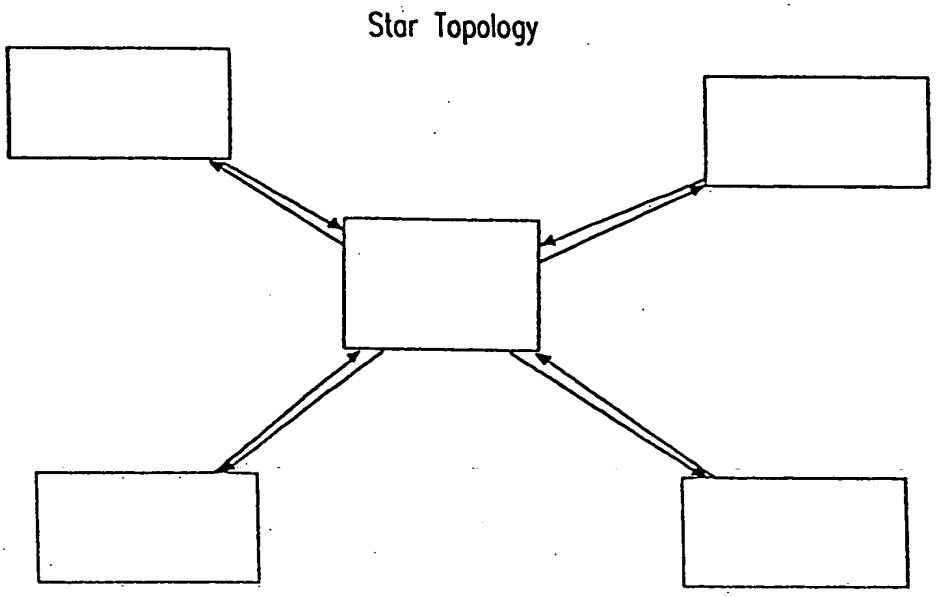

FIG.4

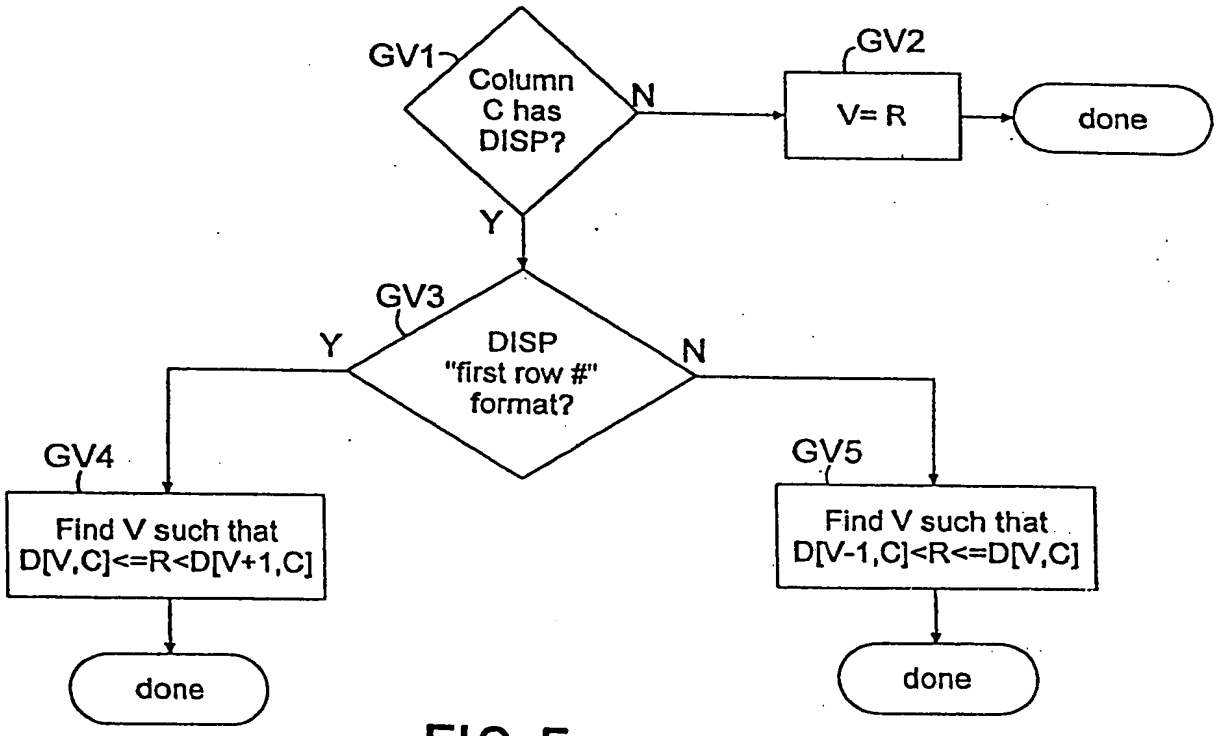
Row get_valrec(Row R, Column C)
GV1
Column C has DISP?
N
GV2
V= R
done
Y
GV3
DISP "first row #" format?
Y
N
GV4
Find V such that D[V,C]<=R<D[V+1,C]
done
GV5
Find V such that D[V-1,C]<R<=D[V,C]
done

FIG.5

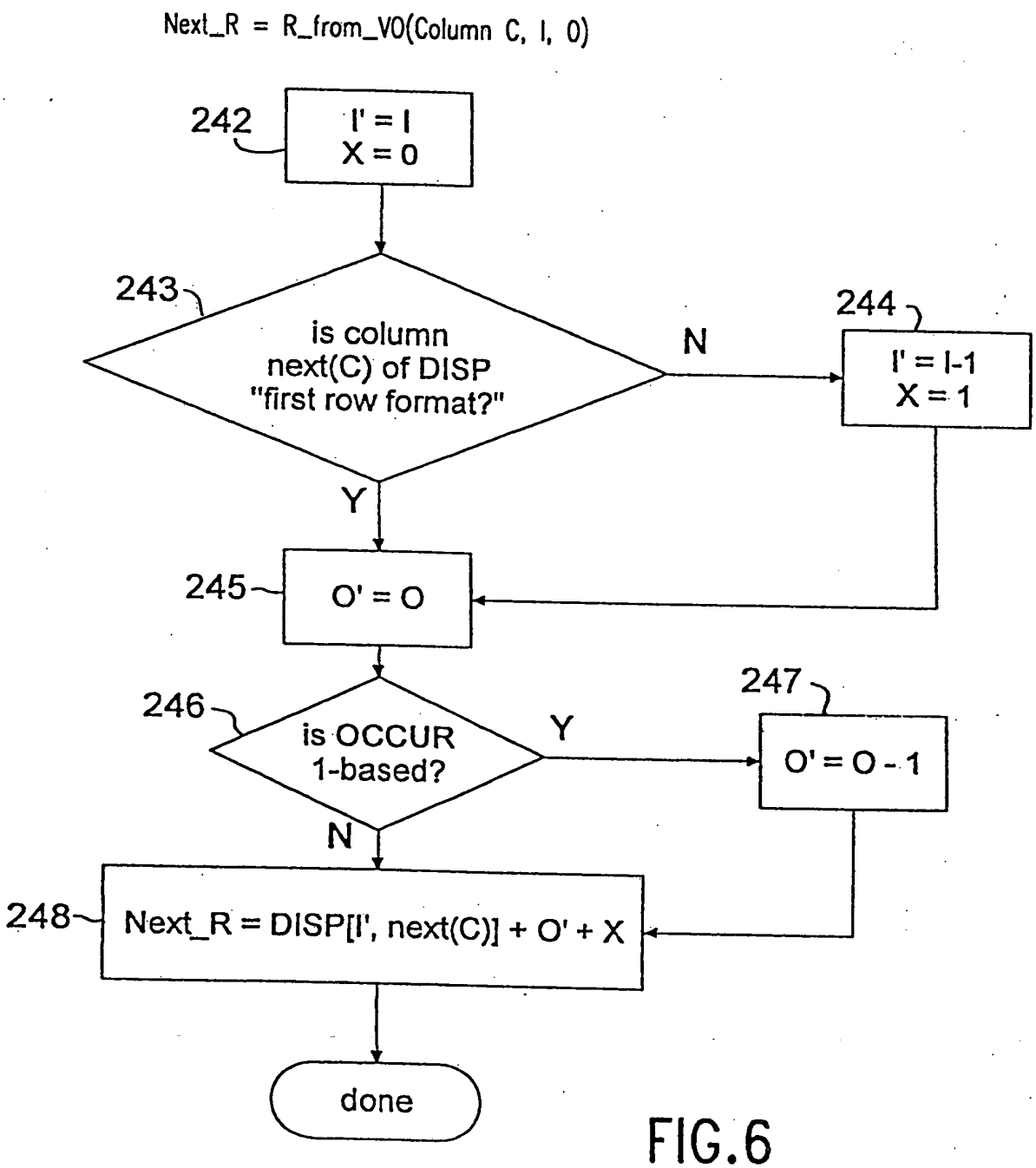
Next_R = R_from_V0(Column C, I, 0)
242
I' = I
X = 0
243
is column next(C) of DISP "first row format?"
N
244
I' = I-1
X = 1
Y
245
O' = O
246
is OCCUR 1-based?
Y
247
O' = O - 1
N
248
Next_R = DISP[I', next(C)] + O' + X
done

FIG.6

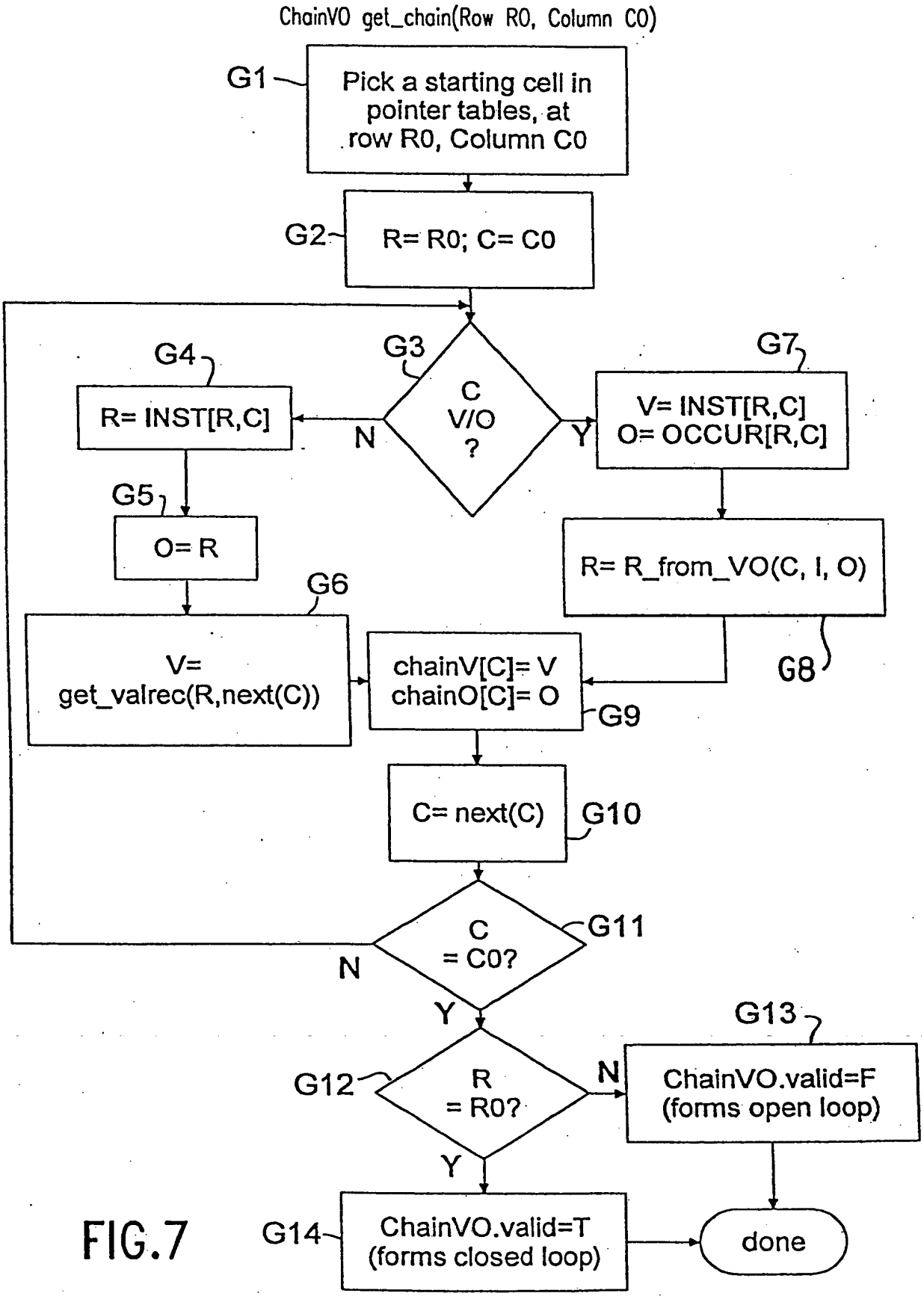
ChainVO get_chain(Row R0, Column C0)
G1
Pick a starting cell in pointer tables, at row R0, Column C0
G2
R= R0; C= C0
G3
C V/O ?
N
Y
G4
R= INST[R,C]
G5
O= R
G6
V= get_valrec(R,next(C))
G7
V= INST[R,C] O= OCCUR[R,C]
R= R_from_VO(C, I, O)
G8
chainV[C]= V chainO[C]= O
G9
C= next(C)
G10
C = C0?
G11
N
Y
G12
R = R0?
N
Y
G13
ChainVO.valid=F (forms open loop)
G14
ChainVO.valid=T (forms closed loop)
done

FIG.7

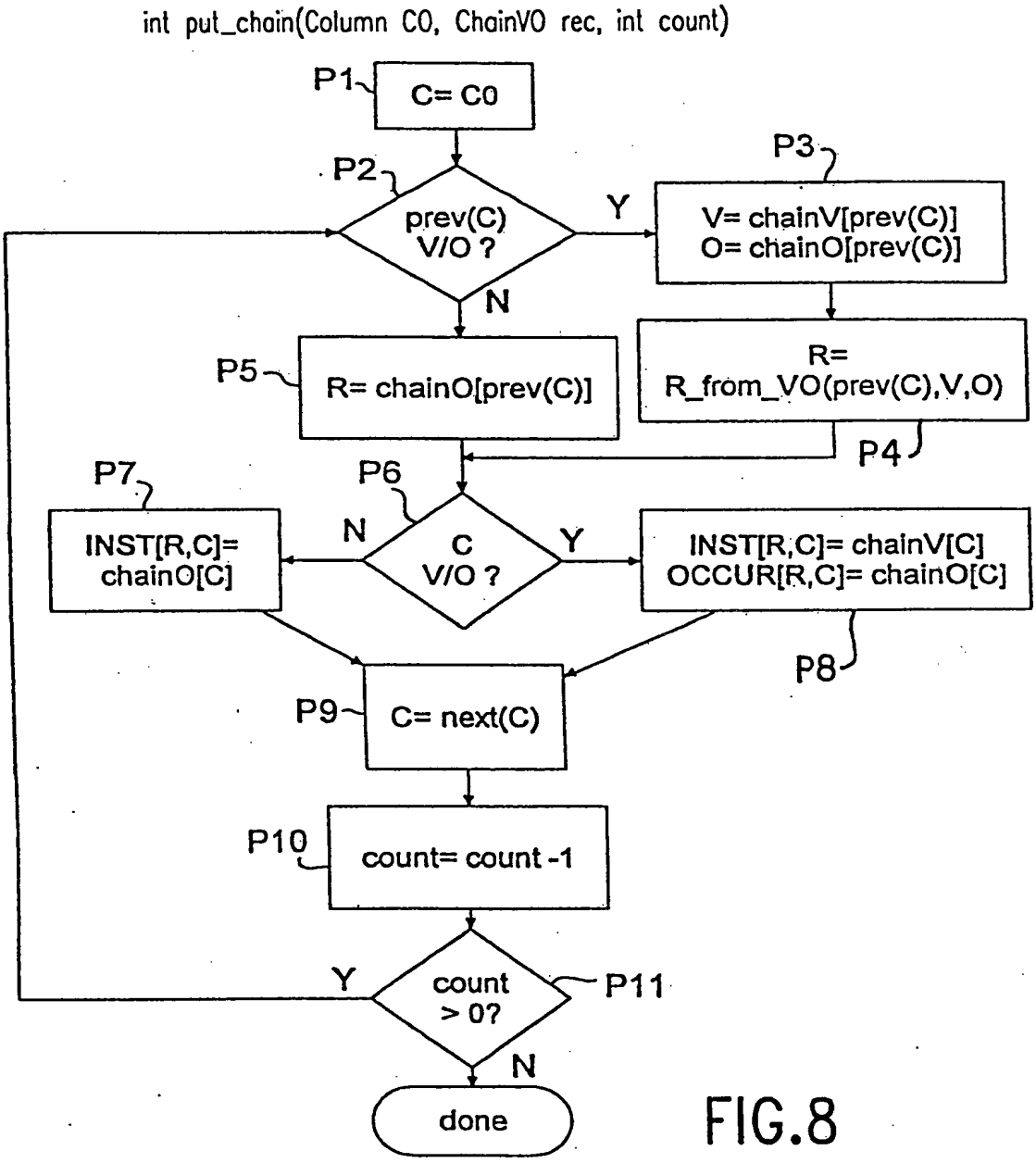
int put_chain(Column C0, ChainVO rec, int count)
P1
C= C0
P2
prev(C) V/O ?
Y
P3
V= chainV[prev(C)]
O= chainO[prev(C)]
N
P5
R= chainO[prev(C)]
R= R_from_VO(prev(C),V,O)
P4
P6
P7
INST[R,C]= chainO[C]
N
C V/O ?
Y
INST[R,C]= chainV[C]
OCCUR[R,C]= chainO[C]
P8
P9
C= next(C)
P10
count= count -1
Y
count > 0?
P11
N
done

FIG.8

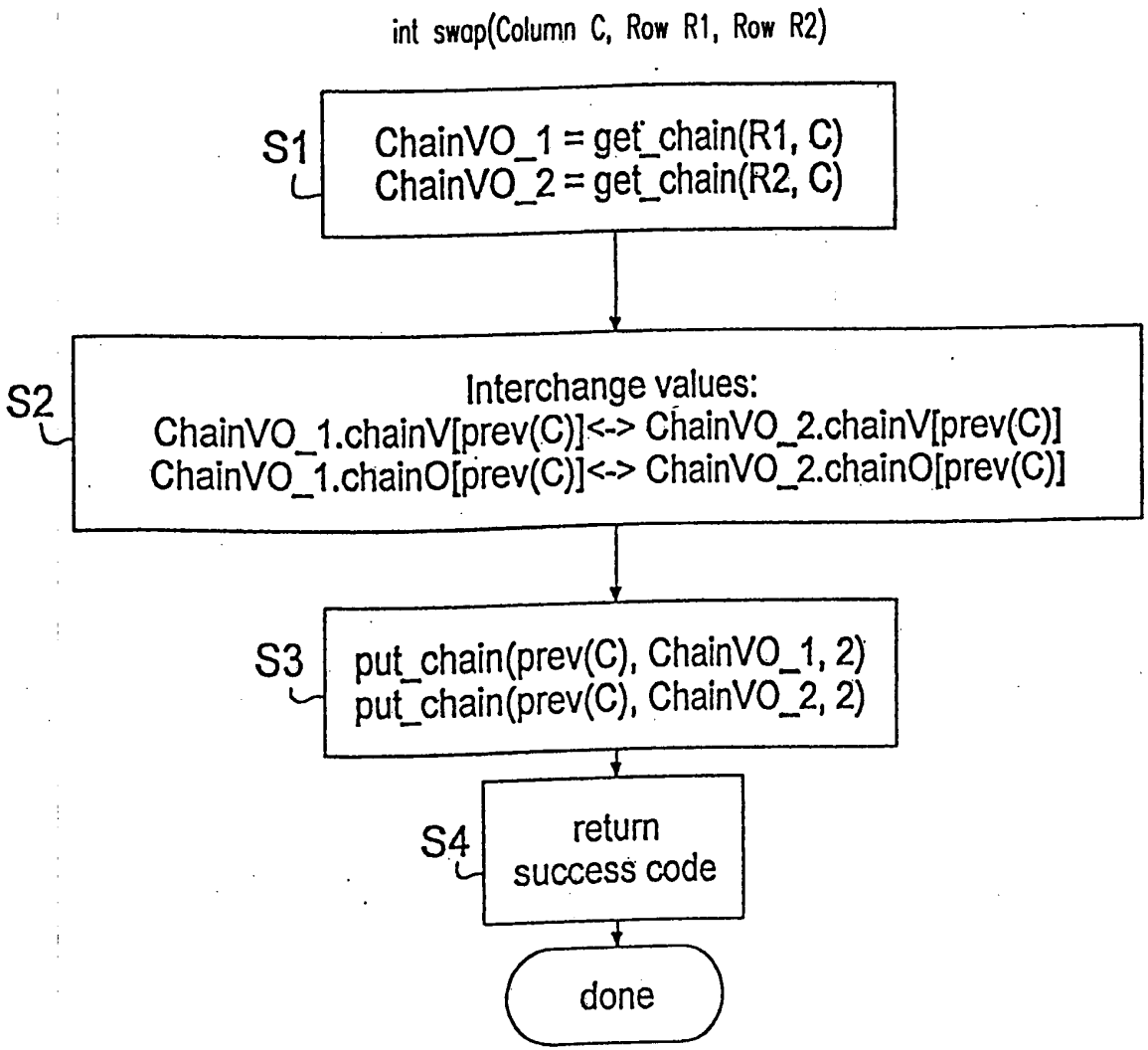
int swap(Column C, Row R1, Row R2)
S1
ChainVO_1 = get_chain(R1, C)
ChainVO_2 = get_chain(R2, C)
S2
Interchange values:
ChainVO_1.chainV[prev(C)]<-> ChainVO_2.chainV[prev(C)]
ChainVO_1.chainO[prev(C)]<-> ChainVO_2.chainO[prev(C)]
S3
put_chain(prev(C), ChainVO_1, 2)
put_chain(prev(C), ChainVO_2, 2)
S4
return
success code
done

FIG.9

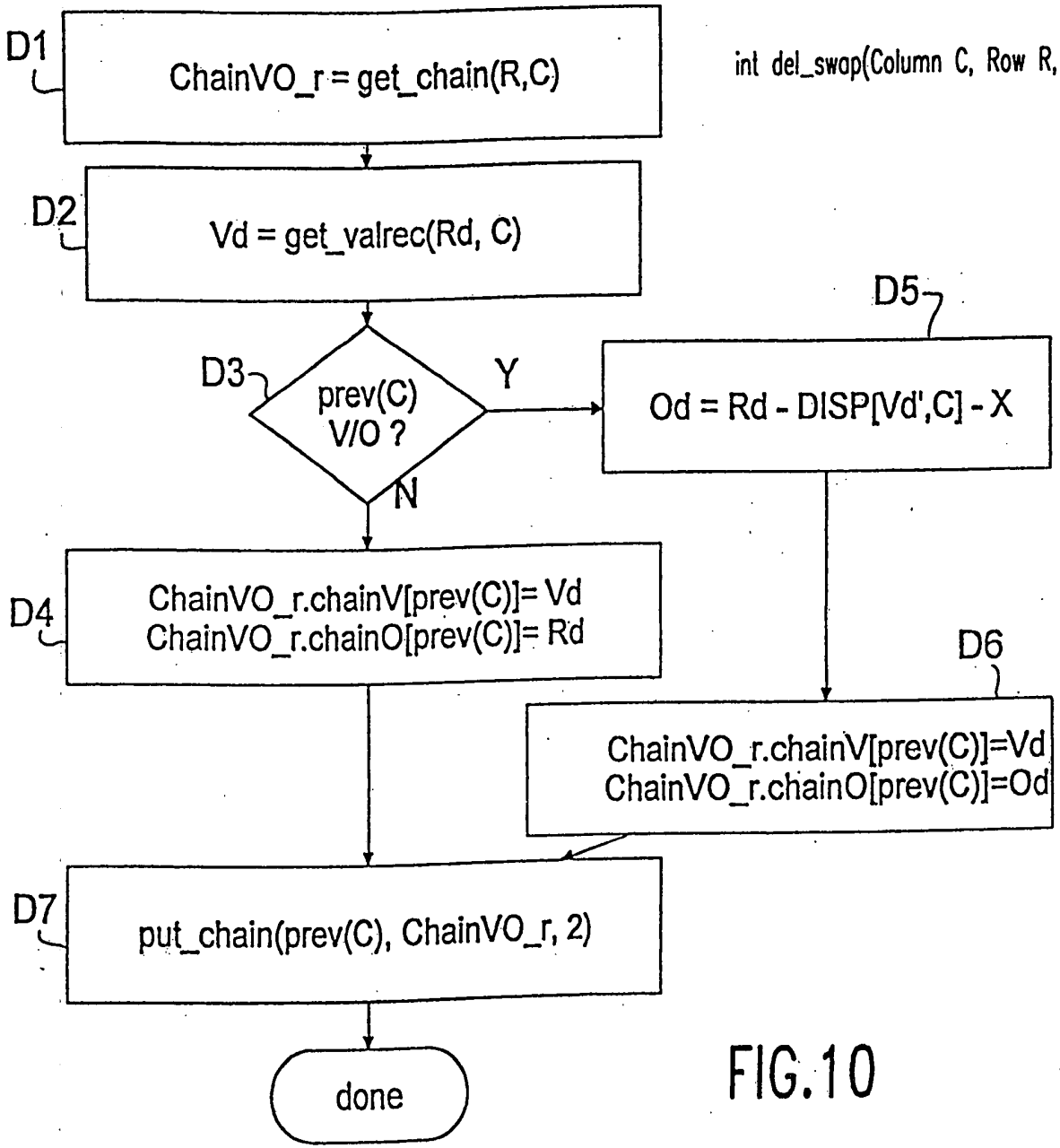
D1
ChainVO_r = get_chain(R,C)
int del_swap(Column C, Row R, Row Rd)
D2
Vd = get_valrec(Rd, C)
D5
D3
prev(C)
V/O ?
Y
Od = Rd - DISP[Vd',C] - X
N
D4
ChainVO_r.chainV[prev(C)]= Vd
ChainVO_r.chainO[prev(C)]= Rd
D6
ChainVO_r.chainV[prev(C)]=Vd
ChainVO_r.chainO[prev(C)]=Od
D7
put_chain(prev(C), ChainVO_r, 2)
done

FIG.10

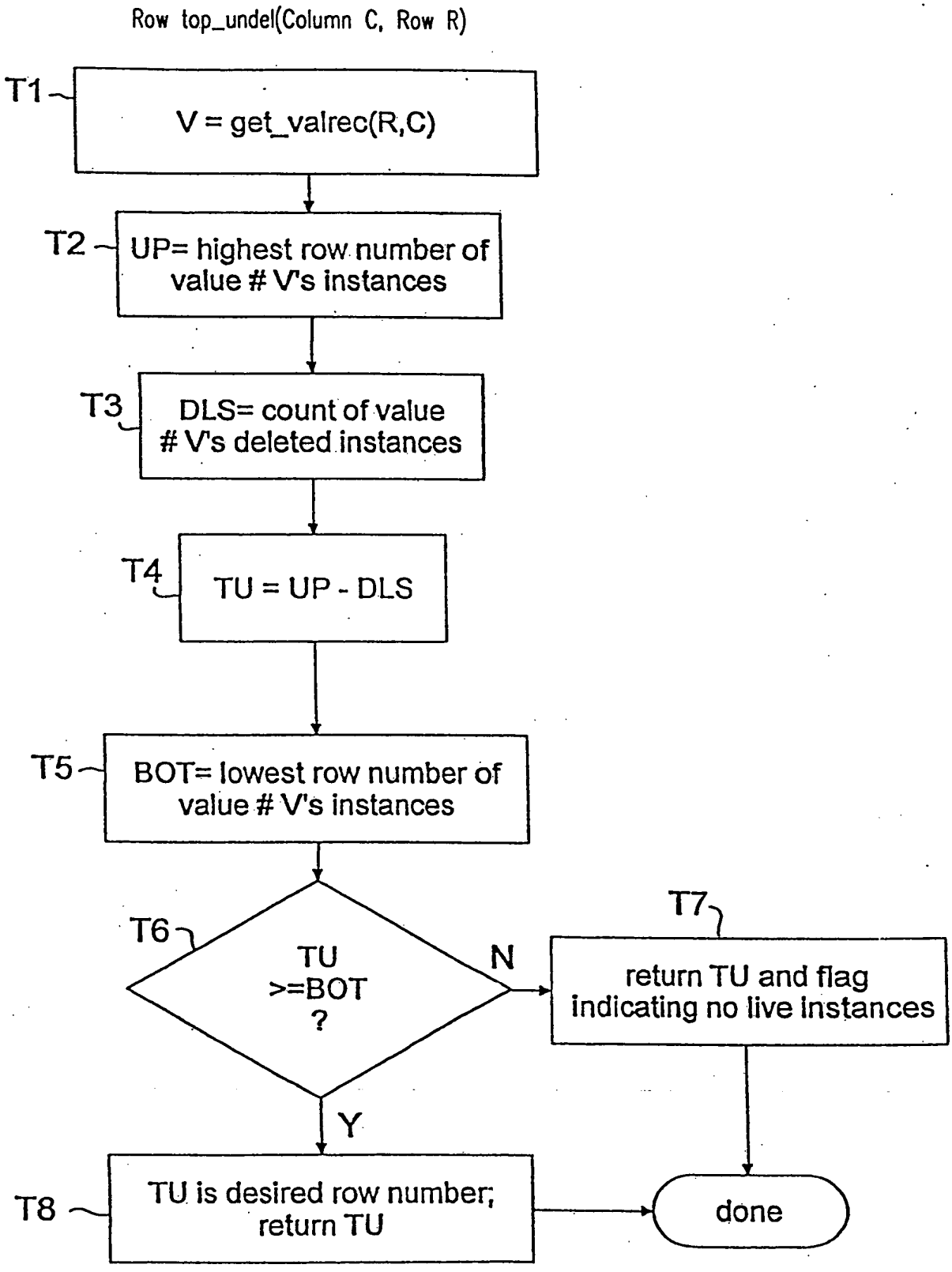
Row top_undel(Column C, Row R)
T1
V = get_valrec(R,C)
T2
UP= highest row number of value # V's instances
T3
DLS= count of value # V's deleted instances
T4
TU = UP - DLS
T5
BOT= lowest row number of value # V's instances
T6
TU >=BOT ?
N
T7
return TU and flag indicating no live instances
Y
T8
TU is desired row number; return TU
done

FIG.11

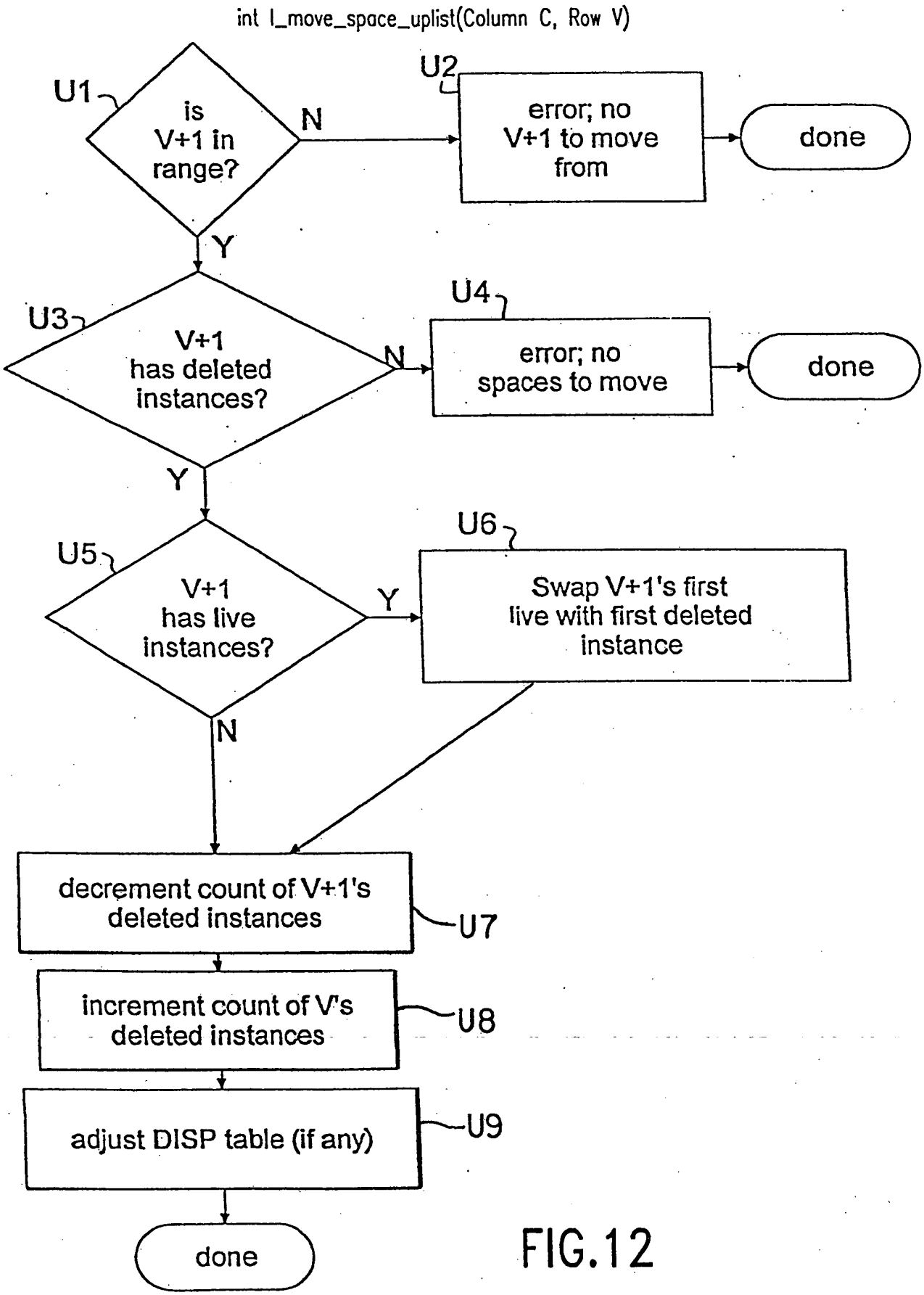
int l_move_space_uplist(Column C, Row V)
U1
is V+1 in range?
N
U2
error; no V+1 to move from
done
Y
U3
V+1 has deleted instances?
N
U4
error; no spaces to move
done
Y
U5
V+1 has live instances?
Y
U6
Swap V+1's first live with first deleted instance
N
decrement count of V+1's deleted instances
U7
increment count of V's deleted instances
U8
adjust DISP table (if any)
U9
done

FIG.12

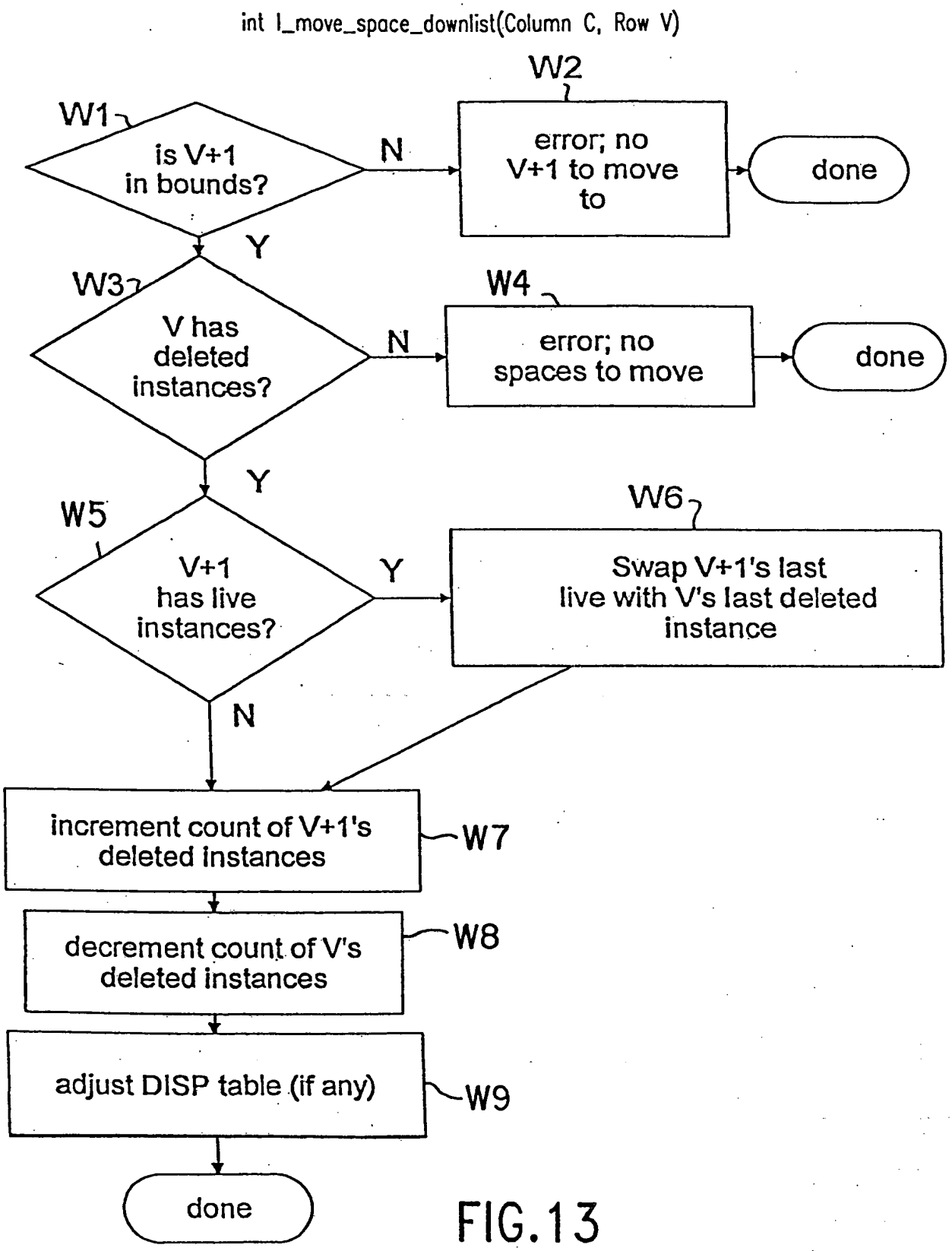
int l_move_space_downlist(Column C, Row V)
W1
is V+1 in bounds?
N
W2
error; no V+1 to move to
done
Y
W3
V has deleted instances?
N
W4
error; no spaces to move
done
Y
W5
V+1 has live instances?
Y
W6
Swap V+1's last live with V's last deleted instance
N
increment count of V+1's deleted instances
W7
decrement count of V's deleted instances
W8
adjust DISP table (if any)
W9
done

FIG.13

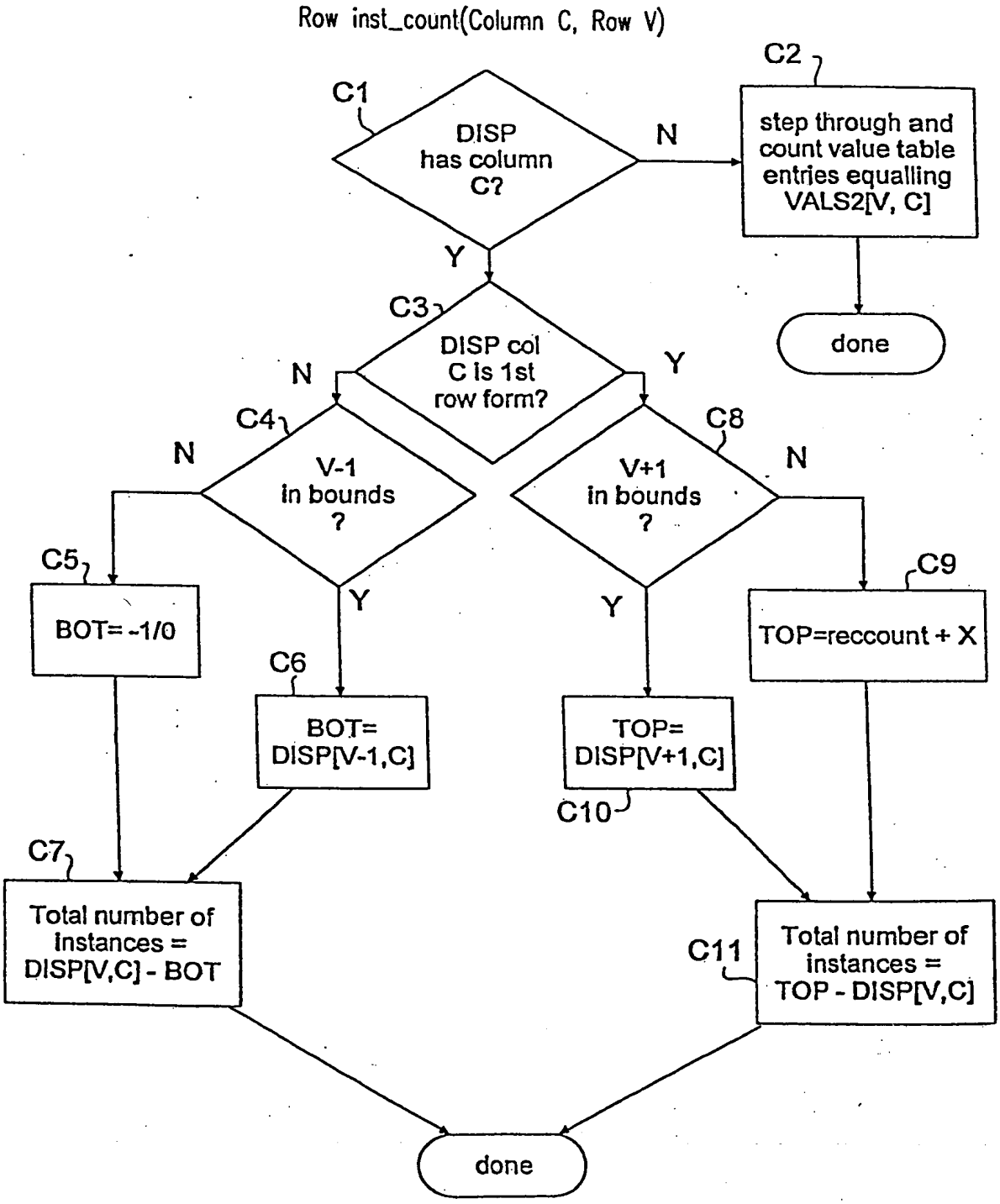
Row inst_count(Column C, Row V)
C1
DISP has column C?
N
C2
step through and count value table entries equalling VALS2[V, C]
done
Y
C3
DISP col C is 1st row form?
N
Y
C4
V-1 in bounds ?
N
C8
V+1 in bounds ?
N
C5
BOT= -1/0
Y
C6
BOT= DISP[V-1,C]
Y
C9
TOP=reccount + X
TOP= DISP[V+1,C]
C10
C7
Total number of instances = DISP[V,C] - BOT
C11
Total number of instances = TOP - DISP[V,C]
done

FIG.14

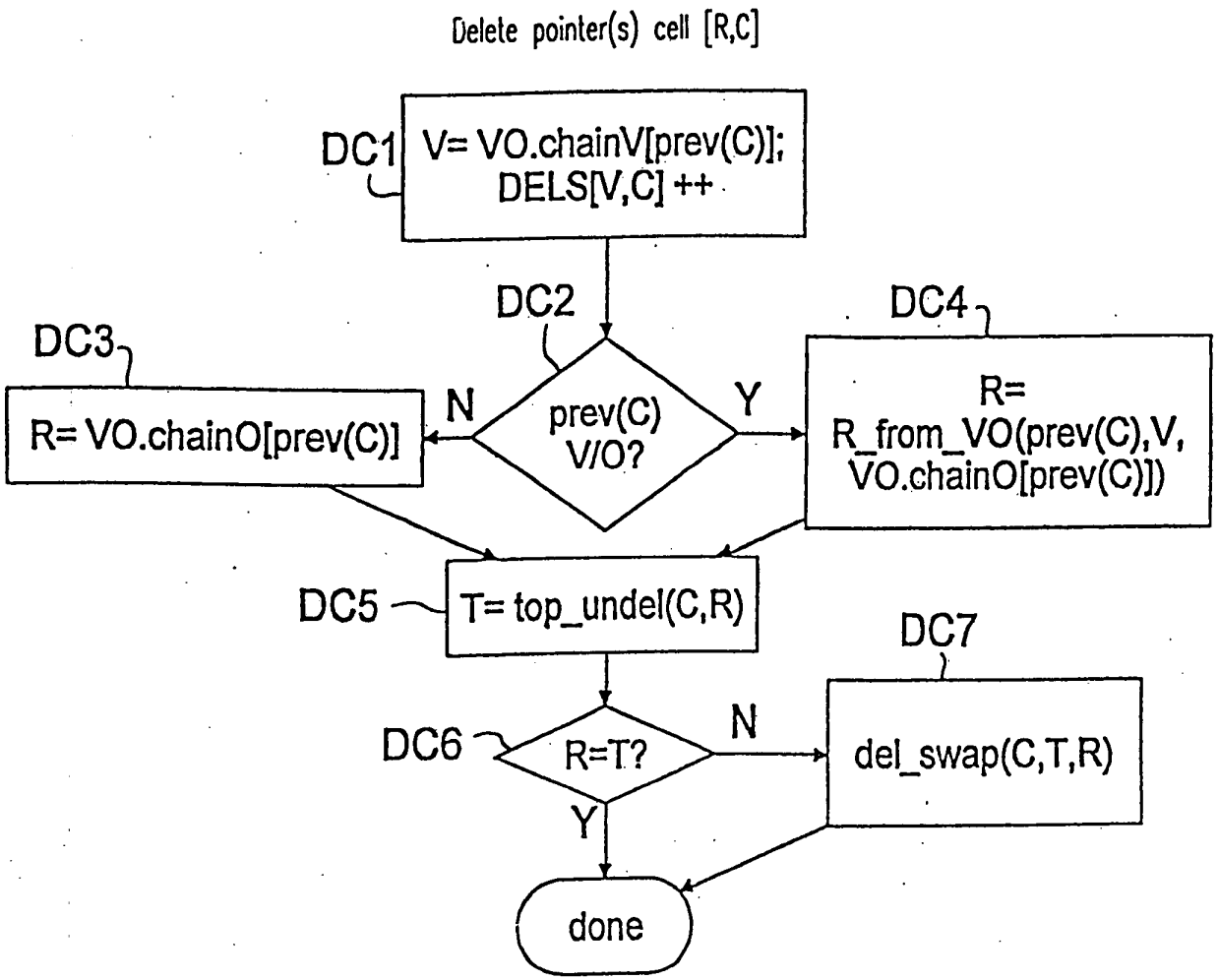
Delete pointer(s) cell [R,C]
DC1
V= VO.chainV[prev(C)];
DELS[V,C] ++
DC2
prev(C)
V/O?
N
Y
DC3
R= VO.chainO[prev(C)]
DC4
R=
R_from_VO(prev(C),V,
VO.chainO[prev(C)])
DC5
T= top_undel(C,R)
DC6
R=T?
N
Y
DC7
del_swap(C,T,R)
done

FIG.15

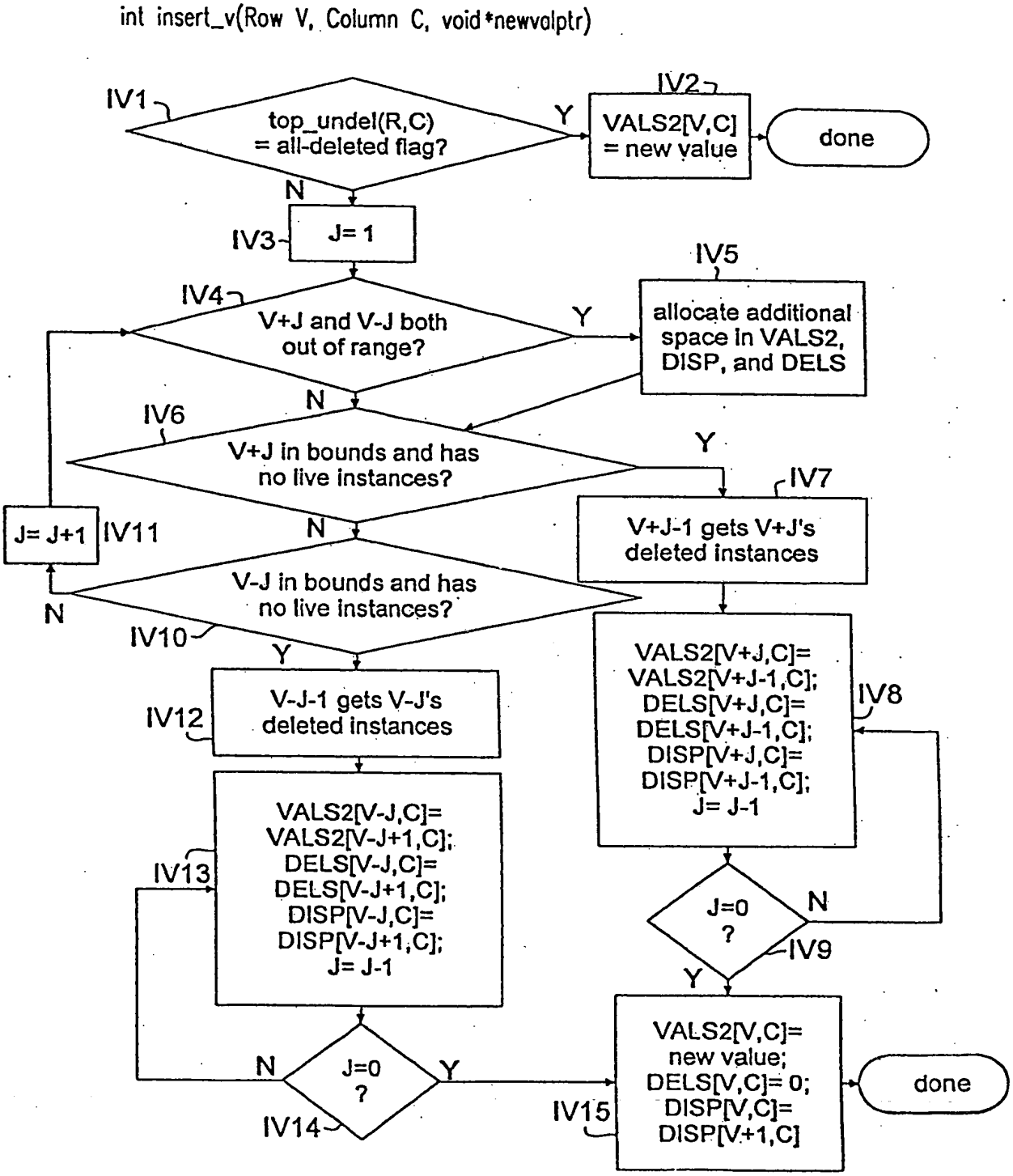
int insert_v(Row V, Column C, void *newvalptr)
IV1
top_undel(R,C) = all-deleted flag?
Y
IV2
VALS2[V,C] = new value
done
N
IV3
J= 1
IV5
IV4
V+J and V-J both out of range?
Y
allocate additional space in VALS2, DISP, and DELS
N
IV6
V+J in bounds and has no live instances?
Y
IV7
V+J-1 gets V+J's deleted instances
J= J+1
IV11
N
V-J in bounds and has no live instances?
N
IV10
Y
VALS2[V+J,C]= VALS2[V+J-1,C]; DELS[V+J,C]= DELS[V+J-1,C]; DISP[V+J,C]= DISP[V+J-1,C]; J= J-1
IV8
V-J-1 gets V-J's deleted instances
IV12
VALS2[V-J,C]= VALS2[V-J+1,C]; DELS[V-J,C]= DELS[V-J+1,C]; DISP[V-J,C]= DISP[V-J+1,C]; J= J-1
IV13
J=0 ?
N
IV9
Y
N
J=0 ?
Y
IV14
VALS2[V,C]= new value; DELS[V,C]= 0; DISP[V,C]= DISP[V+1,C]
IV15
done

FIG.16

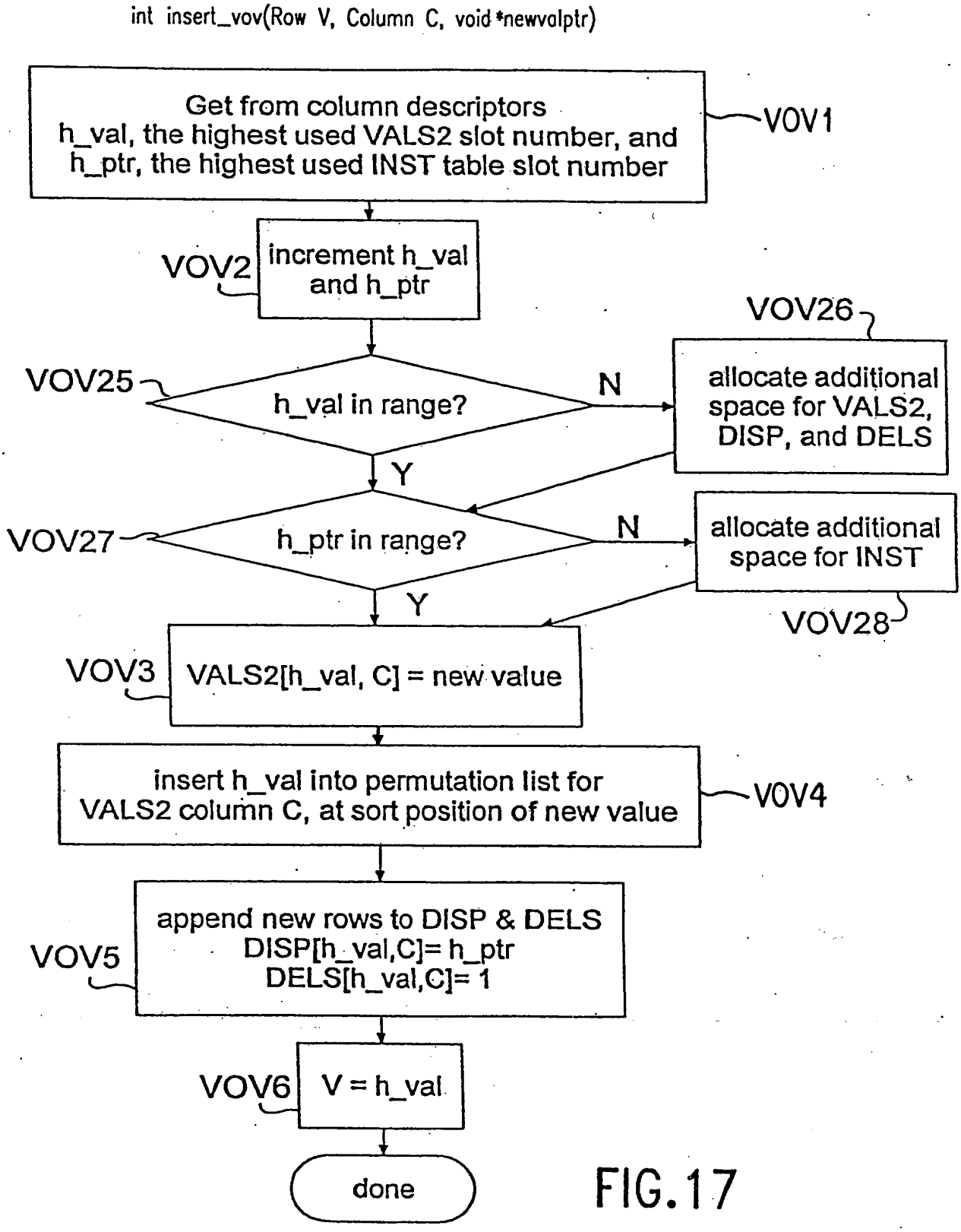
int insert_vov(Row V, Column C, void *newvalptr)
Get from column descriptors h_val, the highest used VALS2 slot number, and h_ptr, the highest used INST table slot number
VOV1
VOV2
increment h_val and h_ptr
VOV26
VOV25
h_val in range?
N
allocate additional space for VALS2, DISP, and DELS
Y
VOV27
h_ptr in range?
N
allocate additional space for INST
Y
VOV28
VOV3
VALS2[h_val, C] = new value
insert h_val into permutation list for VALS2 column C, at sort position of new value
VOV4
append new rows to DISP & DELS
DISP[h_val,C]= h_ptr
DELS[h_val,C]= 1
VOV5
VOV6
V = h_val
done

FIG. 17

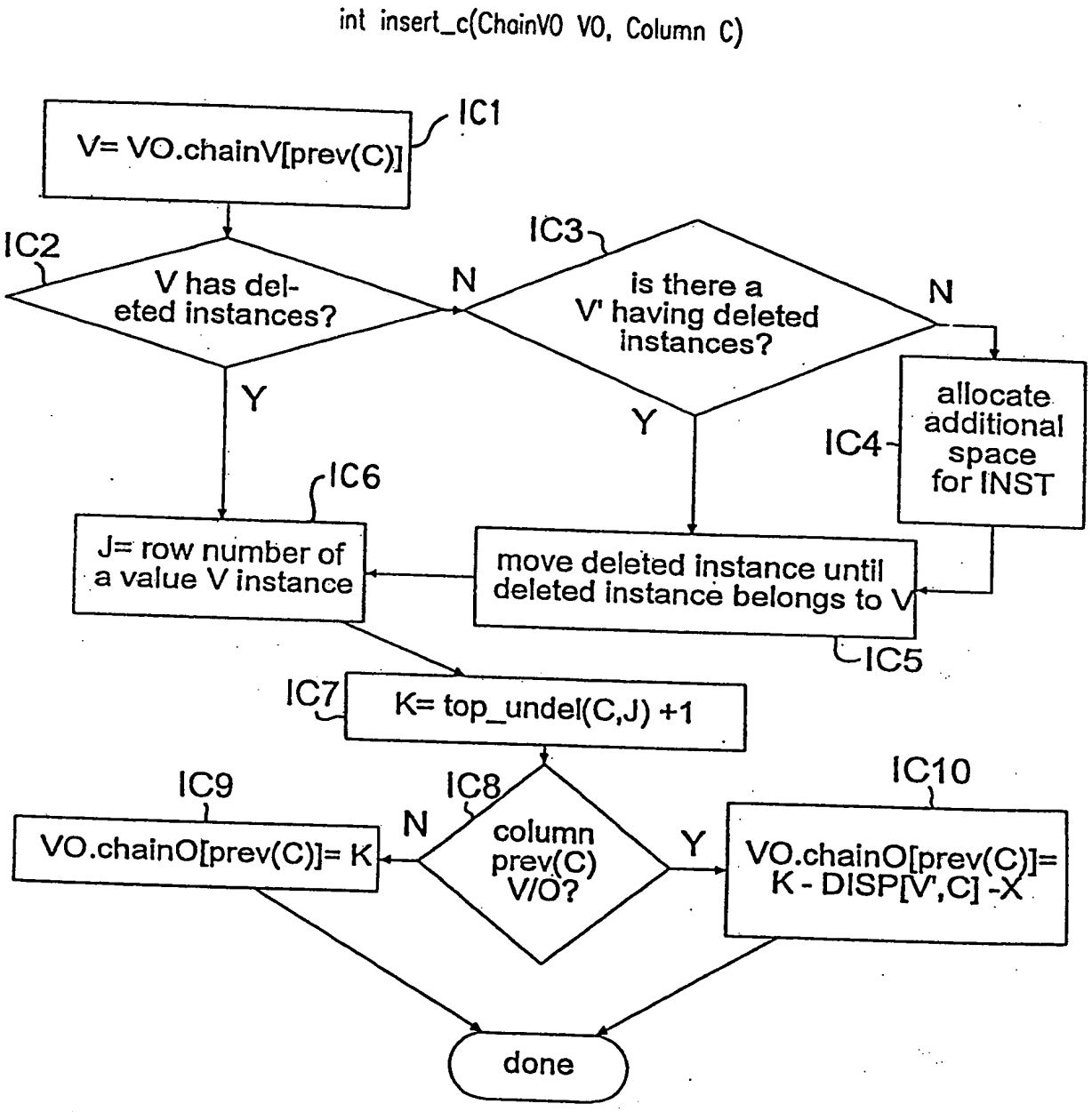
int insert_c(ChainVO VO, Column C)
V= VO.chainV[prev(C)]
IC1
IC2
V has del-eted instances?
N
IC3
is there a V' having deleted instances?
N
Y
Y
IC4
allocate additional space for INST
IC6
J= row number of a value V instance
move deleted instance until deleted instance belongs to V
IC5
IC7
K= top_undel(C,J) +1
IC9
IC8
IC10
VO.chainO[prev(C)]= K
N
column prev(C) V/O?
Y
VO.chainO[prev(C)]= K - DISP[V',C] -X
done

FIG.18

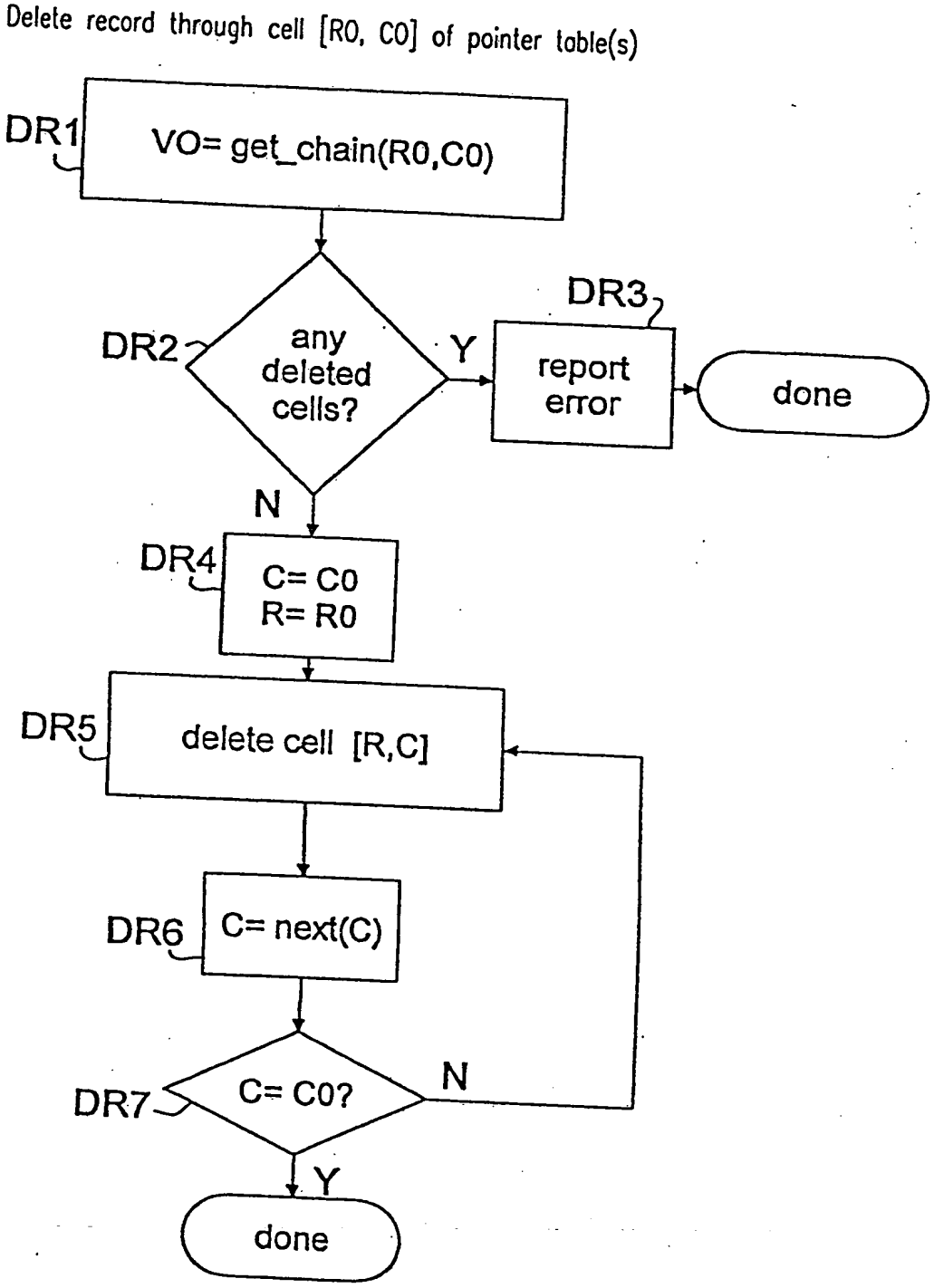
Delete record through cell [R0, C0] of pointer table(s)
DR1
VO= get_chain(R0,C0)
DR2
any deleted cells?
Y
DR3
report error
done
N
DR4
C= C0
R= R0
DR5
delete cell [R,C]
DR6
C= next(C)
DR7
C= C0?
N
Y
done

FIG.19

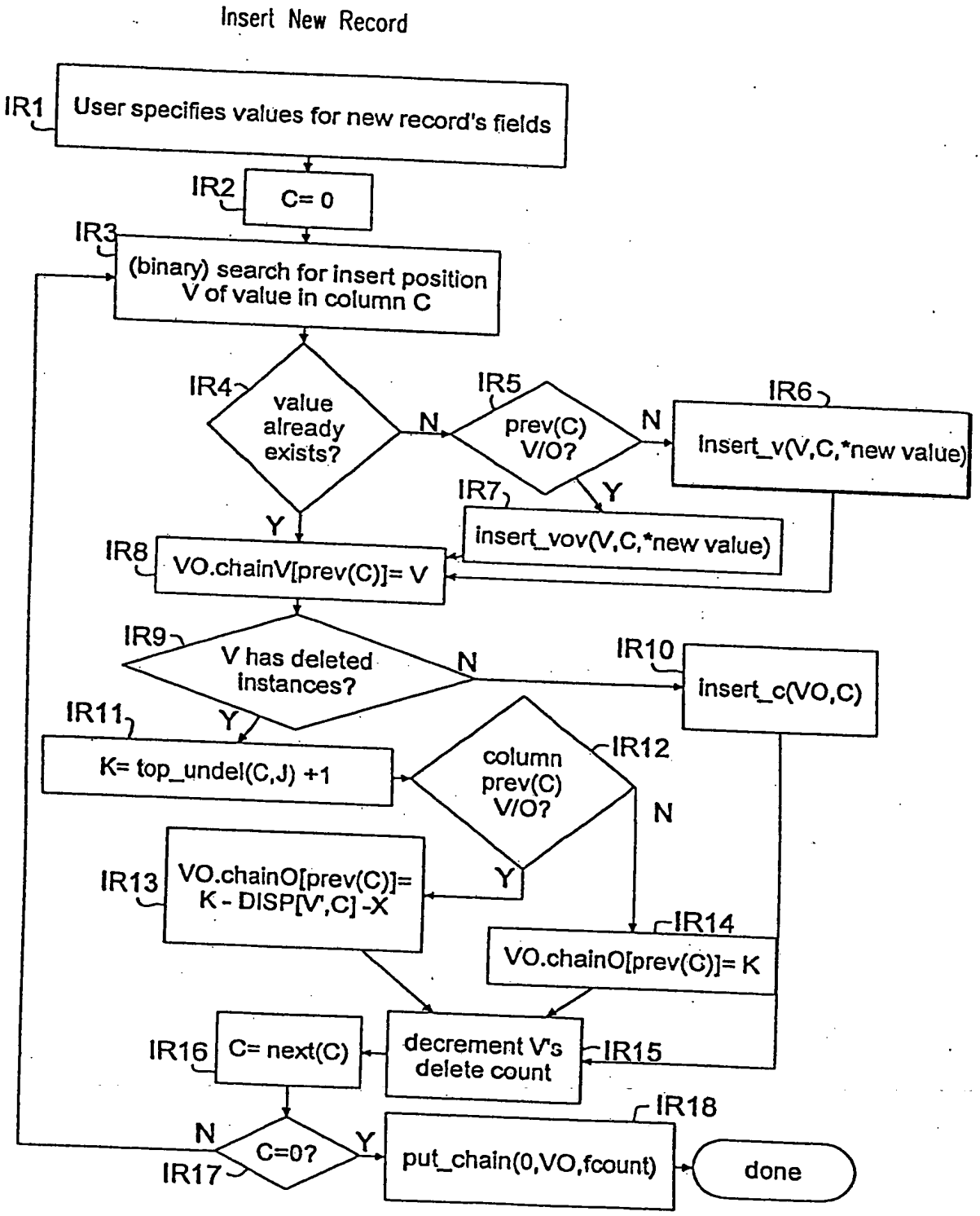
Insert New Record
IR1
User specifies values for new record's fields
IR2
C= 0
IR3
(binary) search for insert position V of value in column C
IR4
value already exists?
N
IR5
prev(C) V/O?
N
IR6
insert_v(V,C,*new value)
Y
IR7
insert_vov(V,C,*new value)
Y
IR8
VO.chainV[prev(C)]= V
IR9
V has deleted instances?
N
IR10
insert_c(VO,C)
Y
IR11
K= top_undel(C,J) +1
column prev(C) V/O?
IR12
N
Y
IR13
VO.chainO[prev(C)]= K - DISP[V',C] -X
IR14
VO.chainO[prev(C)]= K
IR16
C= next(C)
decrement V's delete count
IR15
IR18
N
C=0?
Y
IR17
put_chain(0,VO,fcount)
done

FIG. 20

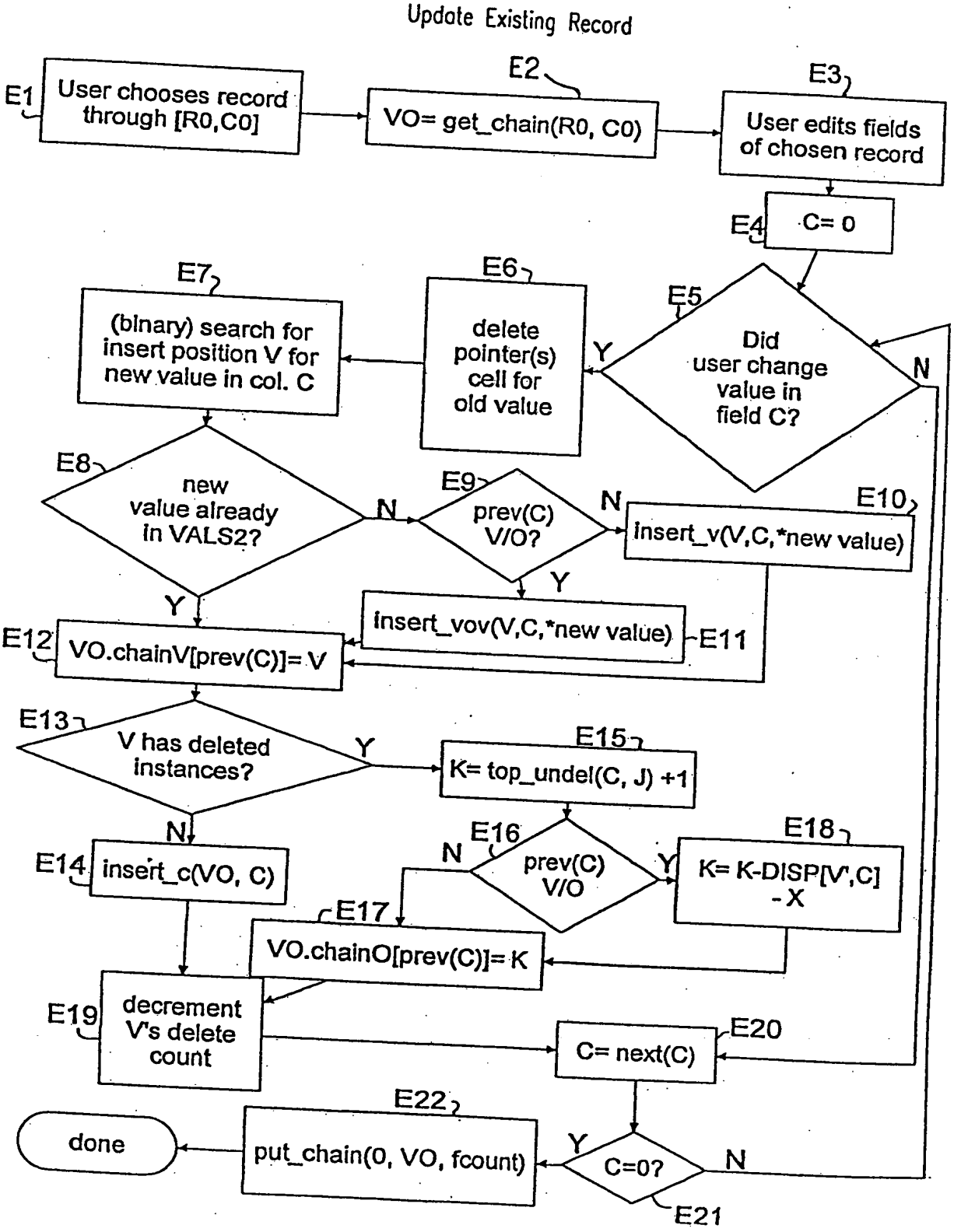
Update Existing Record
E1
User chooses record through [R0,C0]
E2
VO= get_chain(R0, C0)
E3
User edits fields of chosen record
E4
C= 0
E5
Did user change value in field C?
Y
N
E6
delete pointer(s) cell for old value
E7
(binary) search for insert position V for new value in col. C
E8
new value already in VALS2?
N
E9
prev(C) V/O?
N
E10
insert_v(V,C,*new value)
Y
E11
insert_vov(V,C,*new value)
Y
E12
VO.chainV[prev(C)]= V
E13
V has deleted instances?
Y
E15
K= top_undel(C, J) +1
N
E14
insert_c(VO, C)
E16
prev(C) V/O
N
Y
E18
K= K-DISP[V',C] - X
E17
VO.chainO[prev(C)]= K
E19
decrement V's delete count
E20
C= next(C)
E22
done
put_chain(0, VO, fcount)
Y
C=0?
N
E21

FIG.21

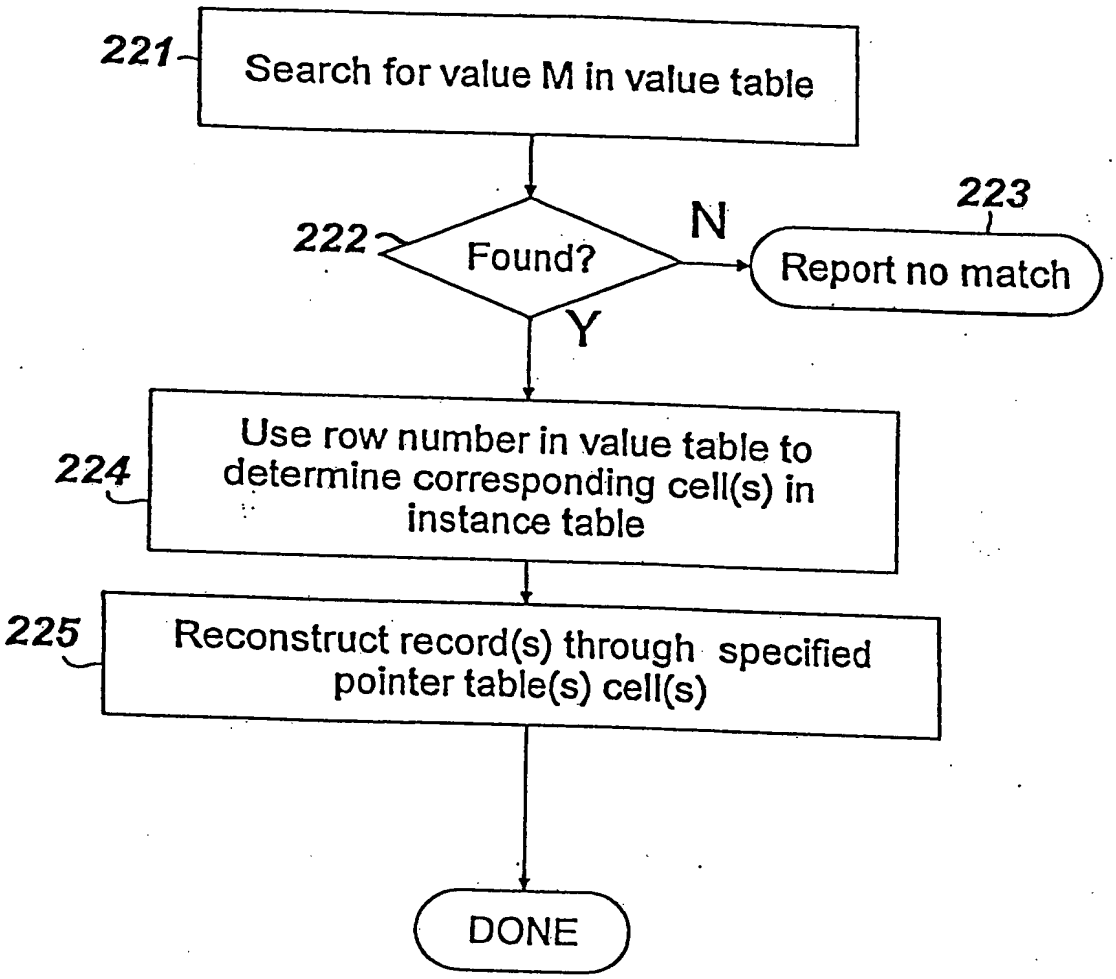
221
Search for value M in value table
222
Found?
N
223
Report no match
Y
224
Use row number in value table to determine corresponding cell(s) in instance table
225
Reconstruct record(s) through specified pointer table(s) cell(s)
DONE

FIG.22

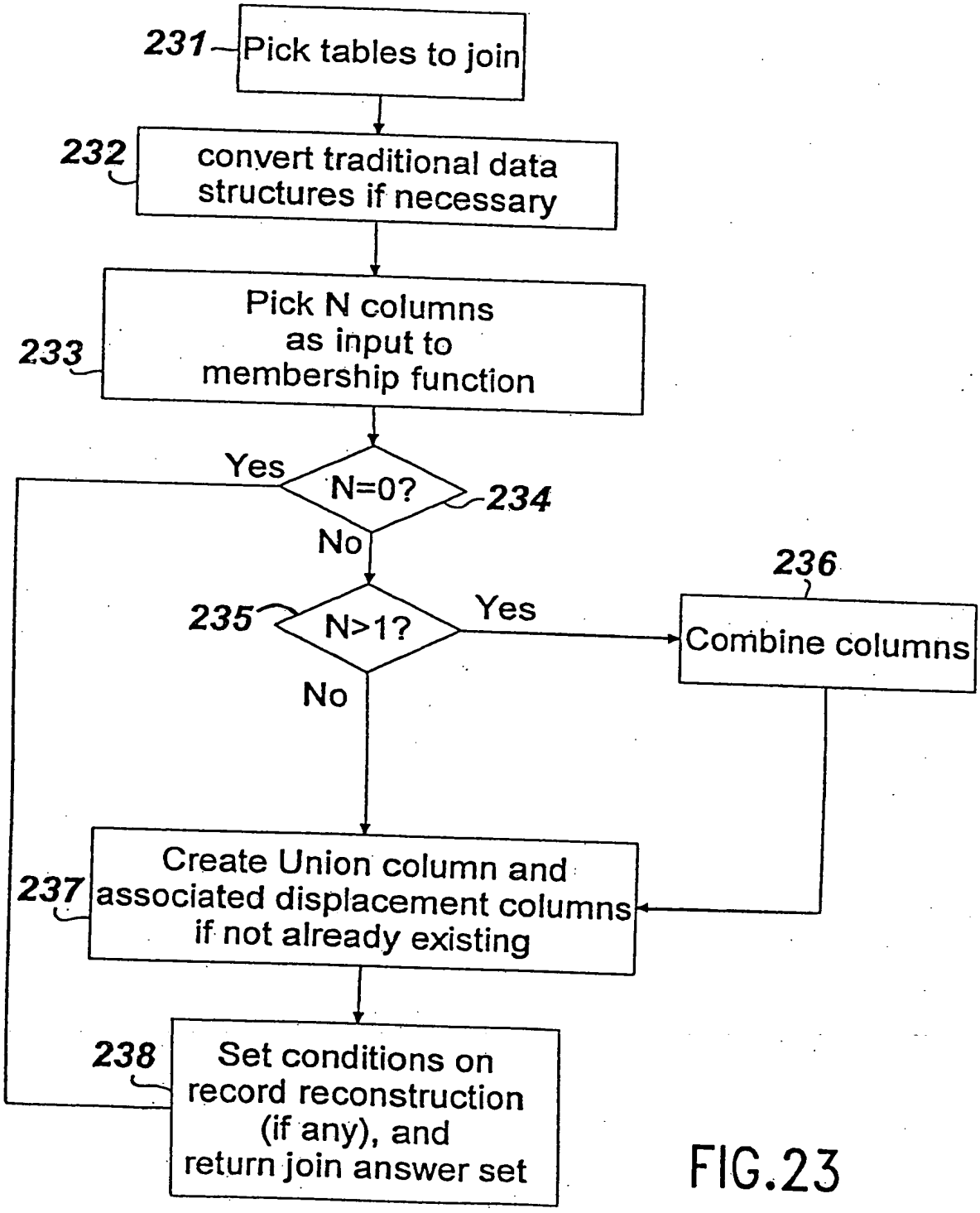
231
Pick tables to join
232
convert traditional data structures if necessary
233
Pick N columns as input to membership function
Yes
N=0?
234
No
235
N>1?
Yes
236
Combine columns
No
237
Create Union column and associated displacement columns if not already existing
238
Set conditions on record reconstruction (if any), and return join answer set

FIG.23